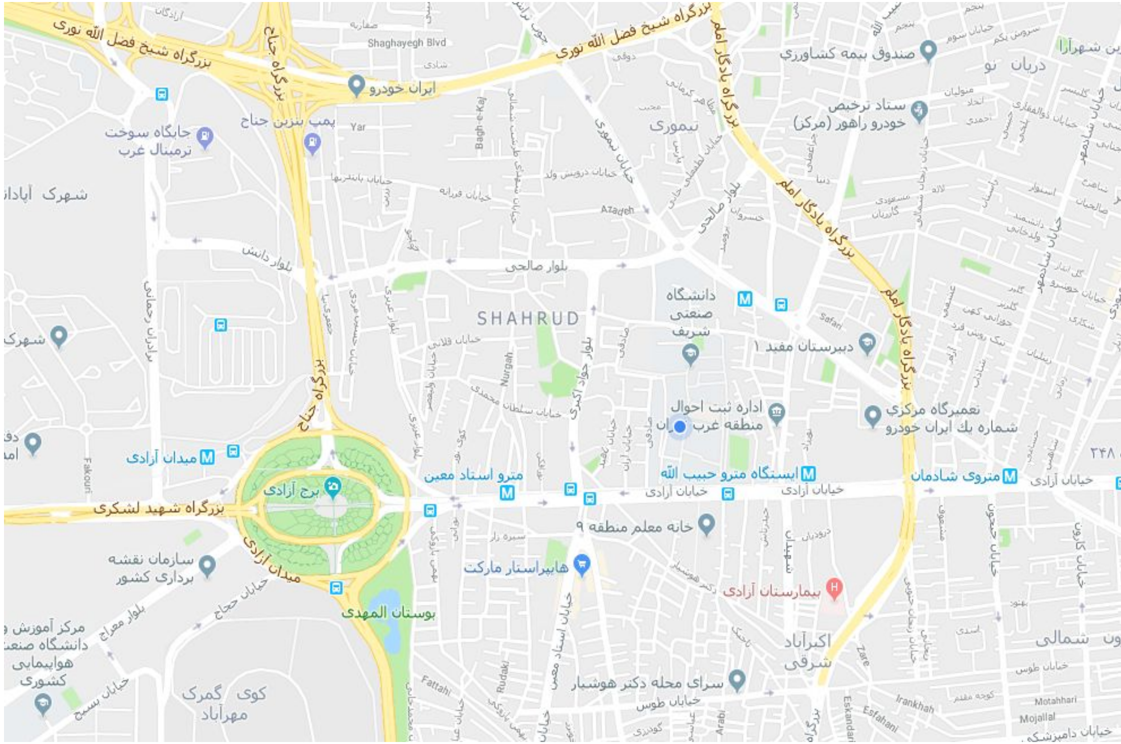
Estimating the Expected Time of Arrival

Ahmad Khajehnejad

Summer 98

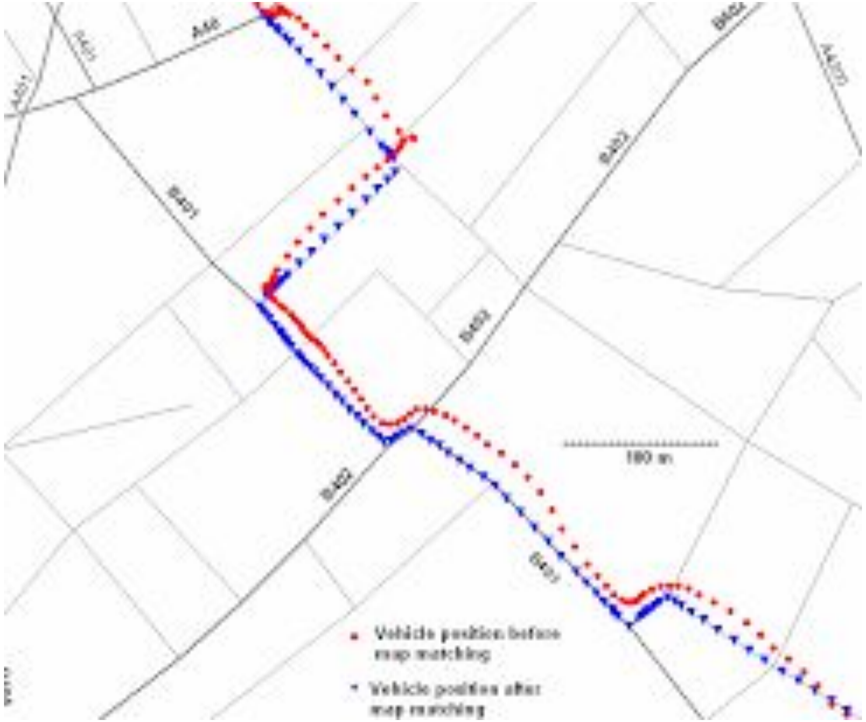
Basic Modules

- Map
- Directed Graph
- How to make:
  - Use third party’s map
  - Extract from a third party
  - Government - Satellite images
  - Use GPS data
  - Manual Edits



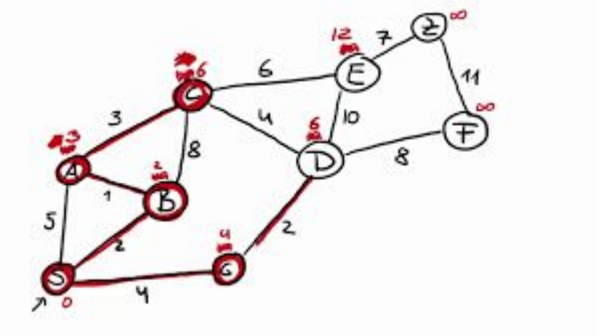
Basic Modules

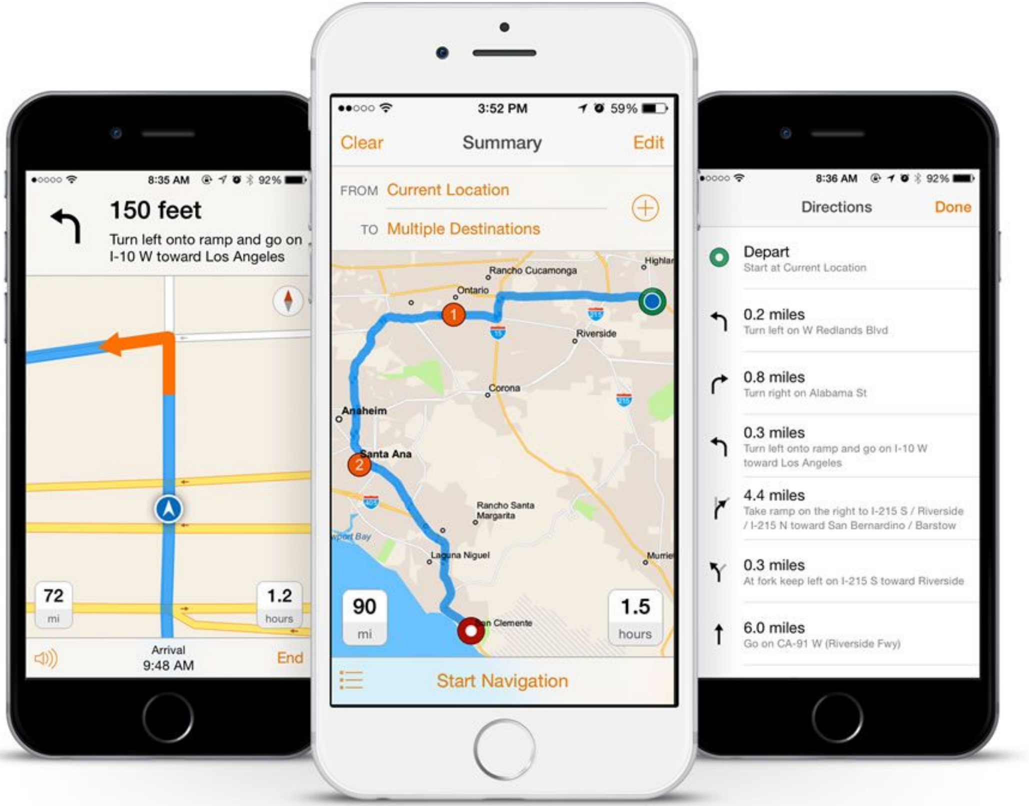
- Map Matcher



Basic Modules

- Routing
- Scale (memory - computation)
- Edge types
- Middle points





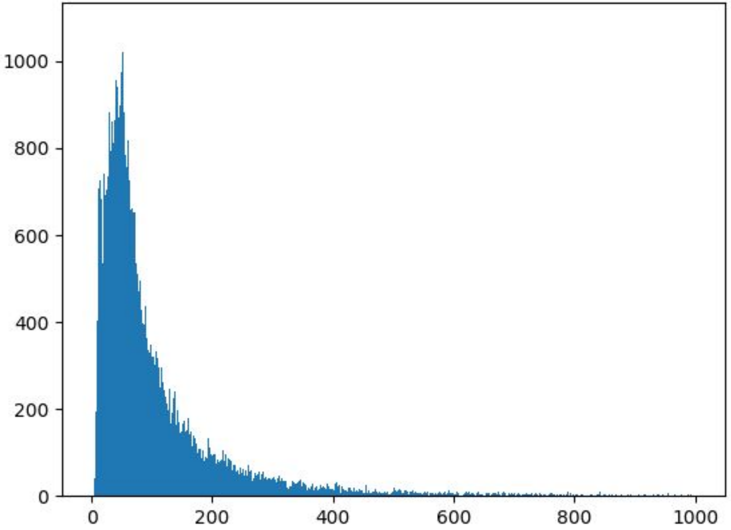
Basic Modules

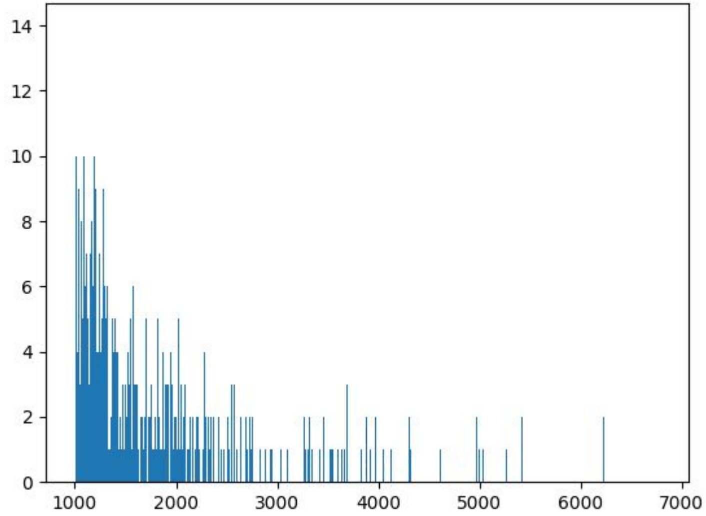
- Expected Time of Arrival (ETA)

- Challenge: few number of active users
- Solution: Asking third parties
    - 3k-4k queries / 15 minutes
    - 300k edges in Tehran
    - T0 and T
    - Prediction and Online

First Order Statistics

- 100k varying edges
- 50k edges varying more than 20%
- Edge types
- Varying edges' lengths

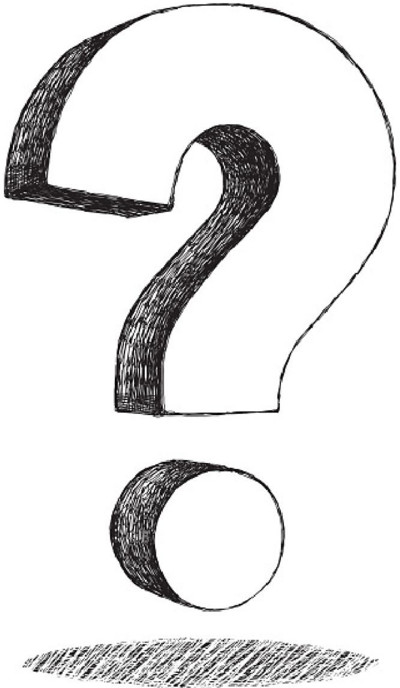




How to overcome the problem?

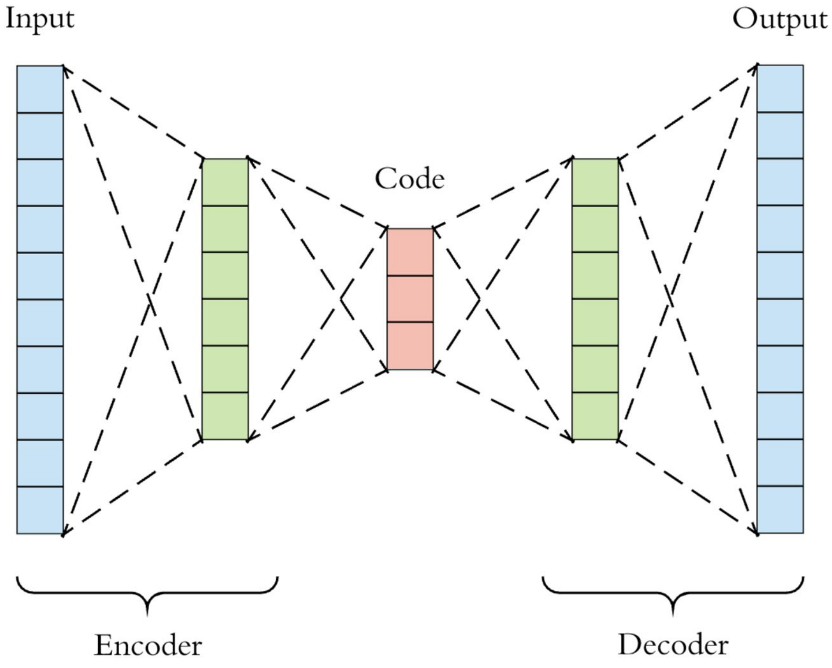
- Find a proper request rate for each edge

- Find a proper request rate for each <edge, time>



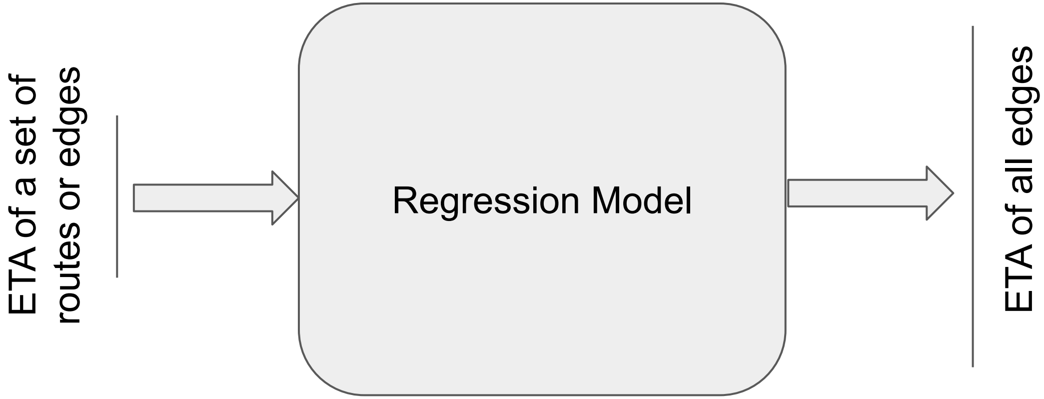
How to overcome the problem?

- Find the state of the traffic
  - Learning an autoencoder

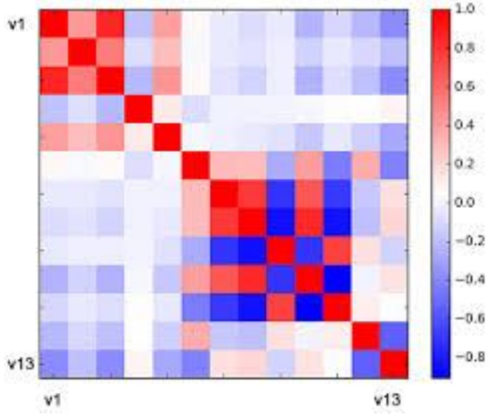


How to overcome the problem?

- Solve a set of regression problems:
  - Ask a set of large routes
  - Ask a subset of edges

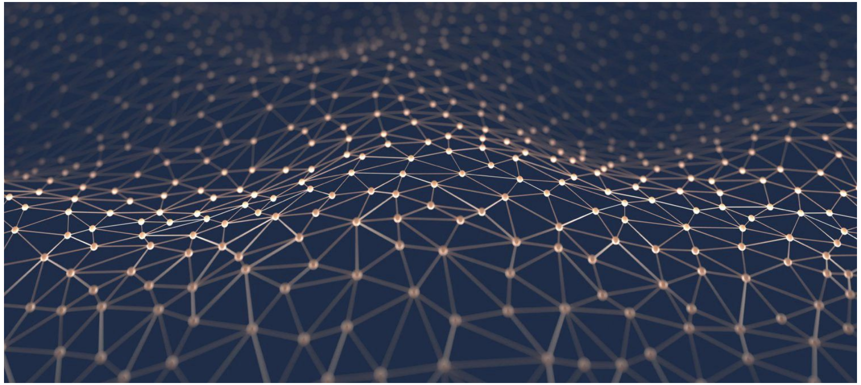


The Regression Problem

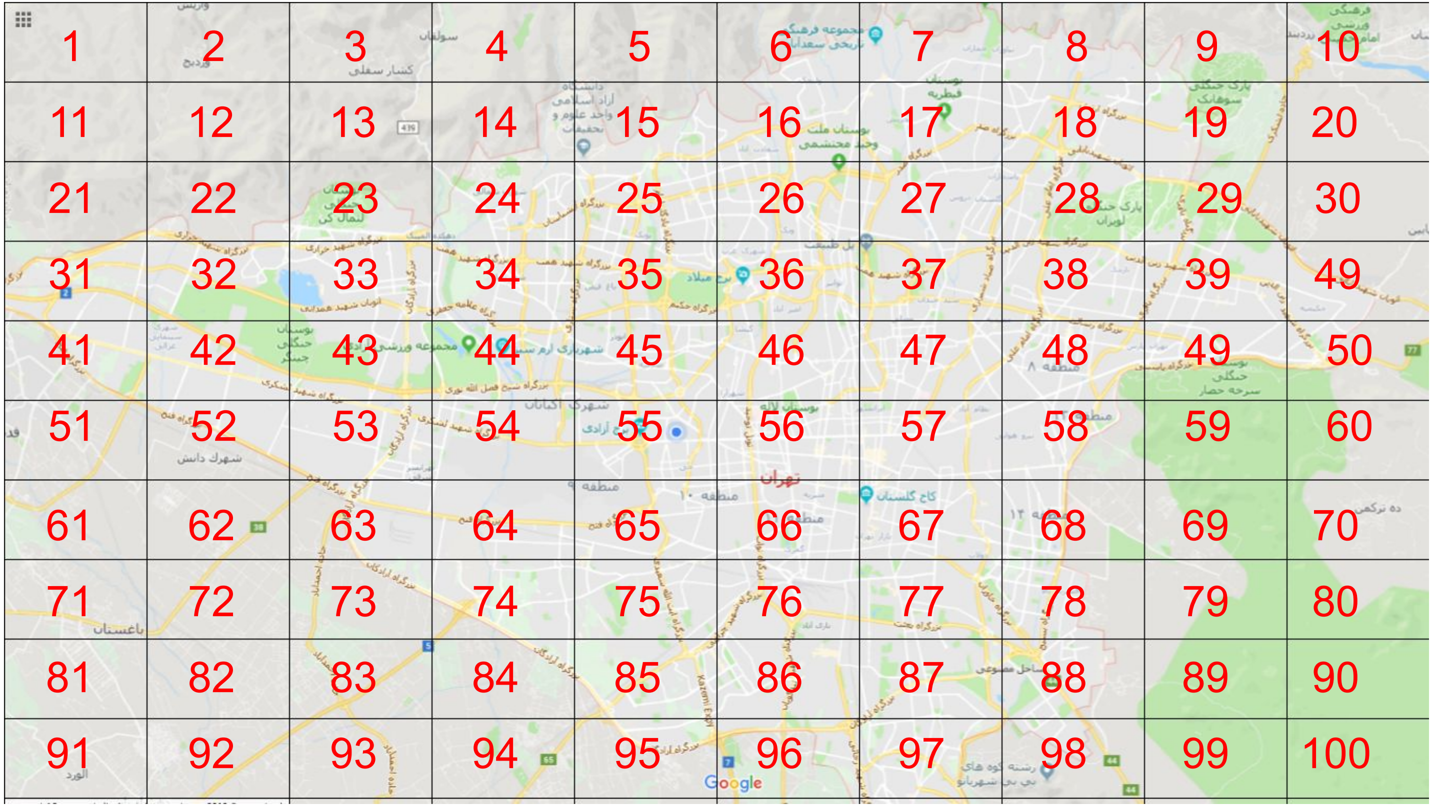


- Finding the correlations
  - Computational problem
  - Linear relations

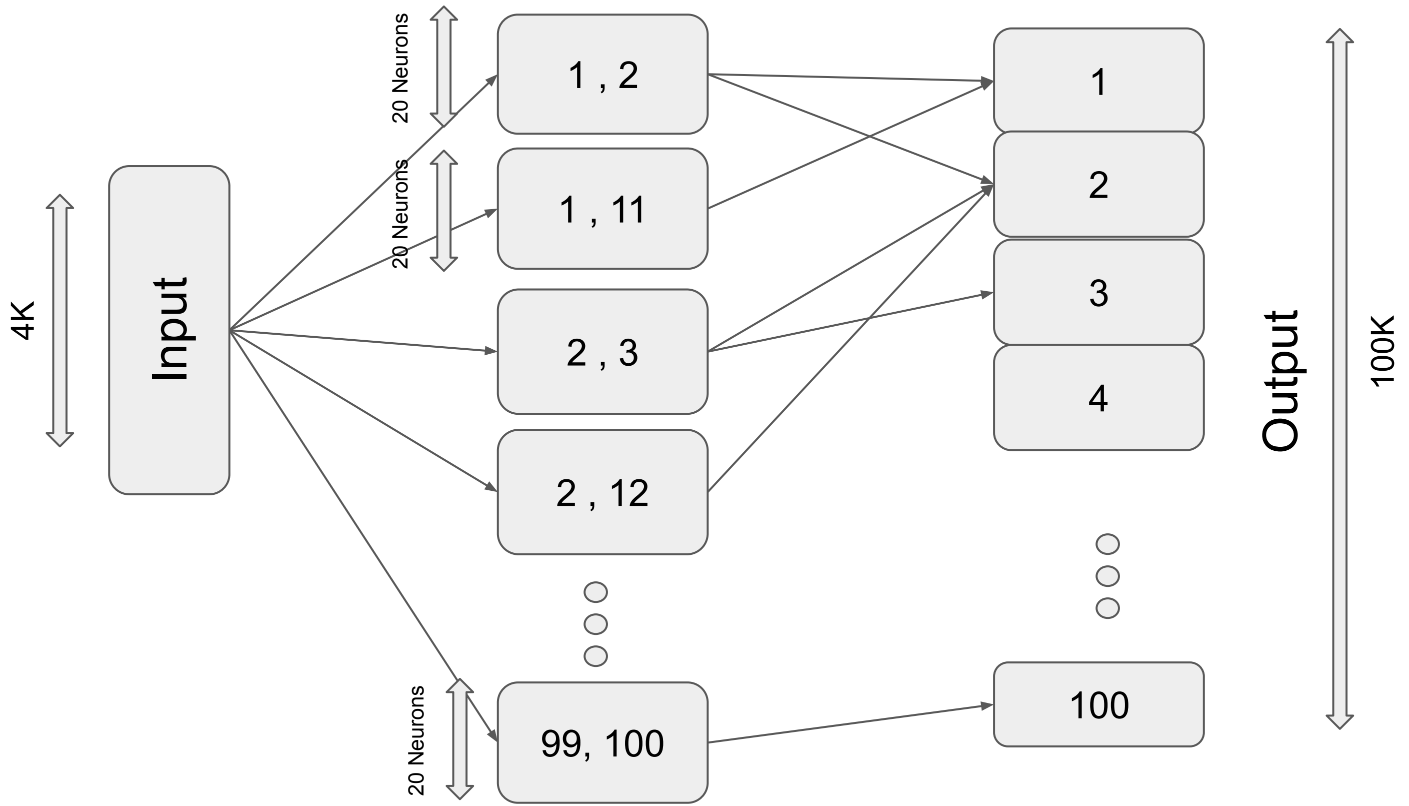
- Using neural networks (easy to solve the optimization problem)
  - structure
  - Input edges
  - Loss function



Neural Network Structure



Neural Network Structure



Input Edges

$$\text{Edge-length} \geq 1000$$

or

$$\text{Edge-length} \geq 120 \quad \& \quad \frac{\text{max-eta}}{\text{min-eta}} \geq 10$$

$$\bigcup$$

1000 most frequent edges

Loss function

$$\frac{\sum_i (T_{pred} - T_{true})^2 \times T_0^{(i)} \times freq^{(i)}}{\sum_i T_0^{(i)} \times freq^{(i)}}$$

Results

80% of <edge-time> pairs ---> error <= 20%

70% of times ---> worst error <= 30%

Q: What KPI to use?

Using GPS data

- GPS fields
    - Edge_id
    - Fraction
    - Fractoin_delta
    - Accuracy
    - Ip
    - Latitude
    - Longitude
    - Provider_is_gps
    - Speed
    - Time
    - Uid
    - act_VEHICLE
    - act_BIKE
    - act_WALKRUN
    - act_STILL
    - act_UNKNOWN
    - act_TILTING
    - act_WALKING
    - act_RUNNING

Dirty data Challenges

- Duplicate data
- Vehicle action is not valid
- Zero speed for moving vehicles
- Incompatibility in time zone
- Time in future

How much data we have

- 30k daily active users

Available data for each edge

- Sessions
- Tracks
- Update the estimations every 5 minutes
- Store the data in Reddis

Weighted Average

Average on duration:

$$\frac{\sum_i w^{(i)} D^{(i)}}{\sum_i w^{(i)}}$$

Average on speed:

$$\frac{\sum_i w^{(i)} V^{(i)}}{\sum_i w^{(i)}}$$

Weighted Average

Average over duration: $$\bar{D} = \frac{\beta e^{-\lambda\Delta_0} T_0 + \alpha \sum_i^n e^{-\lambda\Delta^{(i)}} frac^{(i)} D^{(i)}}{\beta e^{-\lambda\Delta_0} + \alpha \sum_i^n e^{-\lambda\Delta^{(i)}} frac^{(i)}}$$

Average over speed: $$\bar{V} = \frac{\beta e^{-\lambda\Delta_0} V_{T_0} + \alpha \sum_i^n e^{-\lambda\Delta^{(i)}} frac^{(i)} V^{(i)}}{\beta e^{-\lambda\Delta_0} + \alpha \sum_i^n e^{-\lambda\Delta^{(i)}} frac^{(i)}}$$

$$\Delta_{(i)} = \frac{\text{Now} - \text{Arrival time of the } i^{th} \text{ track}}{\text{time Quantum}}$$

Avg Duration vs Avg Velocity

Edge length: 1000 V1: 100 V2: 10

Avg D = (10+100)/2 = 55

Avg V = (100+10)/2 = 55 => estimated D = 1000/55 = 18.18

Avg Duration: sensitive to the fast tracks

Avg Velocity: sensitive to the slow tracks

T0 and Profile

Profile: Adaptive with time, instead of T0

Low Memory Implementation

$$\bar{D}_n = \frac{\beta e^{-\lambda\Delta_0} T_0 + \alpha \sum_i^n e^{-\lambda\Delta^{(i)}} frac^{(i)} D^{(i)}}{\beta e^{-\lambda\Delta_0} + \alpha \sum_i^n e^{-\lambda\Delta^{(i)}} frac^{(i)}}$$

$$A_n = \sum_i^n e^{-\lambda\Delta^{(i)}} frac^{(i)} D^{(i)}$$

$$B_n = \sum_i^n e^{-\lambda\Delta^{(i)}} frac^{(i)}$$

$$\bar{D}_n = \frac{\beta e^{-\lambda\Delta_0} T_0 + \alpha A_n}{\beta e^{-\lambda\Delta_0} + \alpha B_n}$$

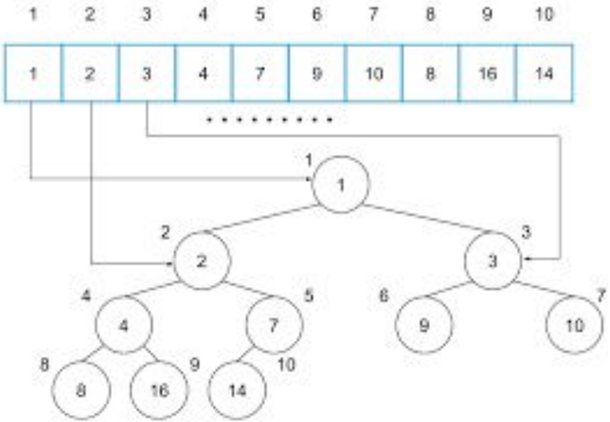
$$A_{n+1} = e^{-\lambda\Delta^{(n+1)}} frac^{(n+1)} D^{(n+1)} + e^{-\lambda\tilde{\Delta}} A_n$$

$$B_{n+1} = e^{-\lambda\Delta^{(n+1)}} frac^{(n+1)} + e^{-\lambda\tilde{\Delta}} B_n$$

$\tilde{\Delta}$: Passed time from last update

Memory less model vs lazy computation

- Other types of coefficients
- The problem of congestion



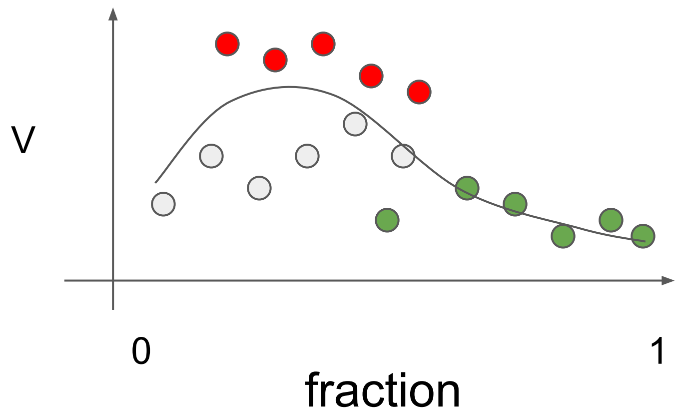
- Using a min Heap (on occurence times of the tracks)
- Bounding a maximum sum of the tracks’ fractions

Weighted Average

Regression point of view:

- Computationally inefficient
- Hard to implement

Segment-based estimation

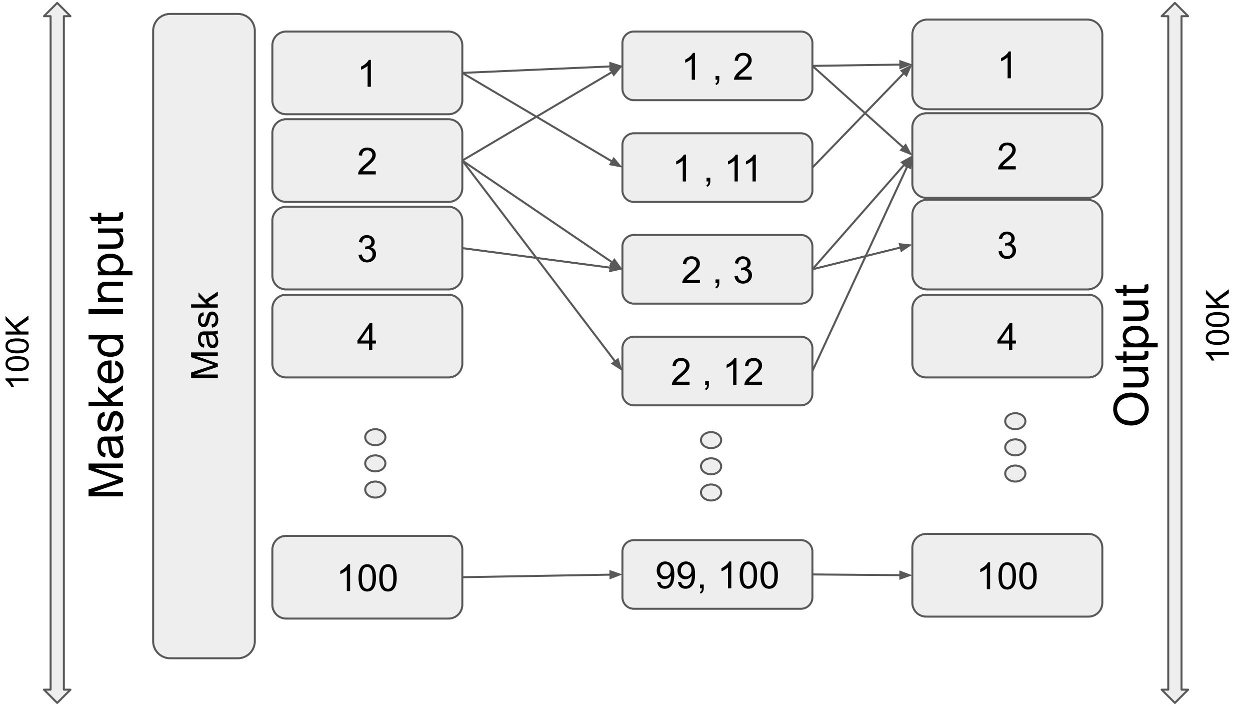


Confidence

$$C = \sum_{i}^{n} e^{-\Theta \Delta^{(i)}} frac^{(i)}$$

Select 1000 most confident edges

Predicting Unconfident Edges



How should the mask be selected? Set the masked inputs to 0 or T0? Pass the confidence into the input layer?

Confident data from third parties

$$\bar{D}_n = \frac{\beta e^{-\lambda\Delta_0} T_0 + \alpha \sum_i^n e^{-\lambda\Delta^{(i)}} frac^{(i)} D^{(i)}}{\beta e^{-\lambda\Delta_0} + \alpha \sum_i^n e^{-\lambda\Delta^{(i)}} frac^{(i)}}$$

Step 1:

$$\Delta_{(i)} = \frac{\text{Now} - \text{Arrival time of the } i^{th} \text{ track}}{\text{time Quantum}}$$

- Receiving limited data (every 15 minutes)
- From heavy traffics
- On a different non-matched map

$\lambda \uparrow$ time Quantum = 15×60

Step 2:

- More clean data

Future: How to use ML more?

- Detecting bad patterns

- End-to-end ETA estimator for one edge

- End-to-end ETA estimator for all the edges

- Recurrent Neural Networks?

Dilemma: Using complex models or domain knowledge?

Future

- Probabilistic estimation
- Learning new profiles
- Traffic prediction
- Routing based on dynamic ETA’s


THANK YOU